Users

CIS 2230 Linux System Administration

Lecture 12

Steve Ruegsegger

Review – shell scripting

- What's the 'proper' first line of a script?
  - What's it called? What does it look like? What does it do?
- What is a “here document”? What does it look like and how is it used?
- How do you find the result (exit status) of a command from the shell prompt?
- What does the test command test for? What is 'true' or 'false'?
- What are the other 2 bash symbols which also “test”?
- What do these do: $( ), $[ ]?
- What is [[ ]] in bash scripting?
- What are the variable names for shell script positional arguments?

Introduction to Users

- Anyone using a Linux computer is a user
- Users' files are stored in /home

```
$ ls /home
steve/   fred/   wilma/     barney/
```

- The primary purpose of separate users is security
  - both from each other as well as for the system itself
- Remember,
  - (Old) Windows/DOS didn't have users
  - The OS assumed the single user should install whatever they wanted.
  - This is what allowed 'bad guys' to deceptively install malware.
- Linux/Unix was always a multi-user system where the OS and users are protected from each other.

Groups

- Users can belong to groups.
- This allows security to be managed for collections of people with different requirements.
- We'll see later that files have one user and one group assigned to them.
- Users can certainly be in multiple groups
- Examples: departments, management, projects, etc.

userids

- Linux keeps track by user IDs
  - User IDs (uid) are numbers (integers)
  - The root user always has uid 0
  - Daemons are (often) system IDs that are less than 100.
  - Users often start at 100 (sometimes 1000).
  - For Ubuntu, users typically start at 1000 and go up by one
  - The id command returns the uid:

```
$ id -u
1000
```

Original. (Not Ubuntu today)

0:	The superuser
1-10:	Daemons and pseudo users
11-99:	System, reserved and "famous" users
100+:	Normal users
60001:	"nobody" (occasionally 32000 or 65534)
60002:	"noaccess" (occasionally 32001)

groups

- There are also group IDs (gid).
- They are a different set of integers that group similar users together.

```
$ id -g
1000

$ groups
steve adm dialout fax cdrom floppy tape dip video
plugdev fuse lpadmin admin sambashare vboxusers

$ id
uid=1000(steve) gid=1000(steve)
groups=4(adm),20(dialout),21(fax),24(cdrom),25(floppy),
26(tape),30(dip),44(video),46(plugdev),104(fuse),105(lp
admin),119(admin),122(sambashare),126(vboxusers),1000(s
teve)
```

adding a user

- https://help.ubuntu.com/community/AddUsersHowto
- To add a new user - use the `adduser` command.

```
$ sudo adduser barney
Adding user `barney' ...
Adding new group `barney' (1005) ...
Adding new user `barney' (1004) with group `barney' ...
Creating home directory `/home/barney' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for barney
Enter the new value, or press ENTER for the default
        Full Name []: Barney
        Room Number []: Rubble
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n]
$ ls /home
barney  cis2230  fred  steve  wilma
```

What exactly did `$adduser` script user do?

- Script, you say? `$ head $(which adduser)`
- What's the user ID?
- Where did that user info go?
- Where's the home directory?
- What files are in the user's home directory? Where did they come from?

Compare adduser to useradd

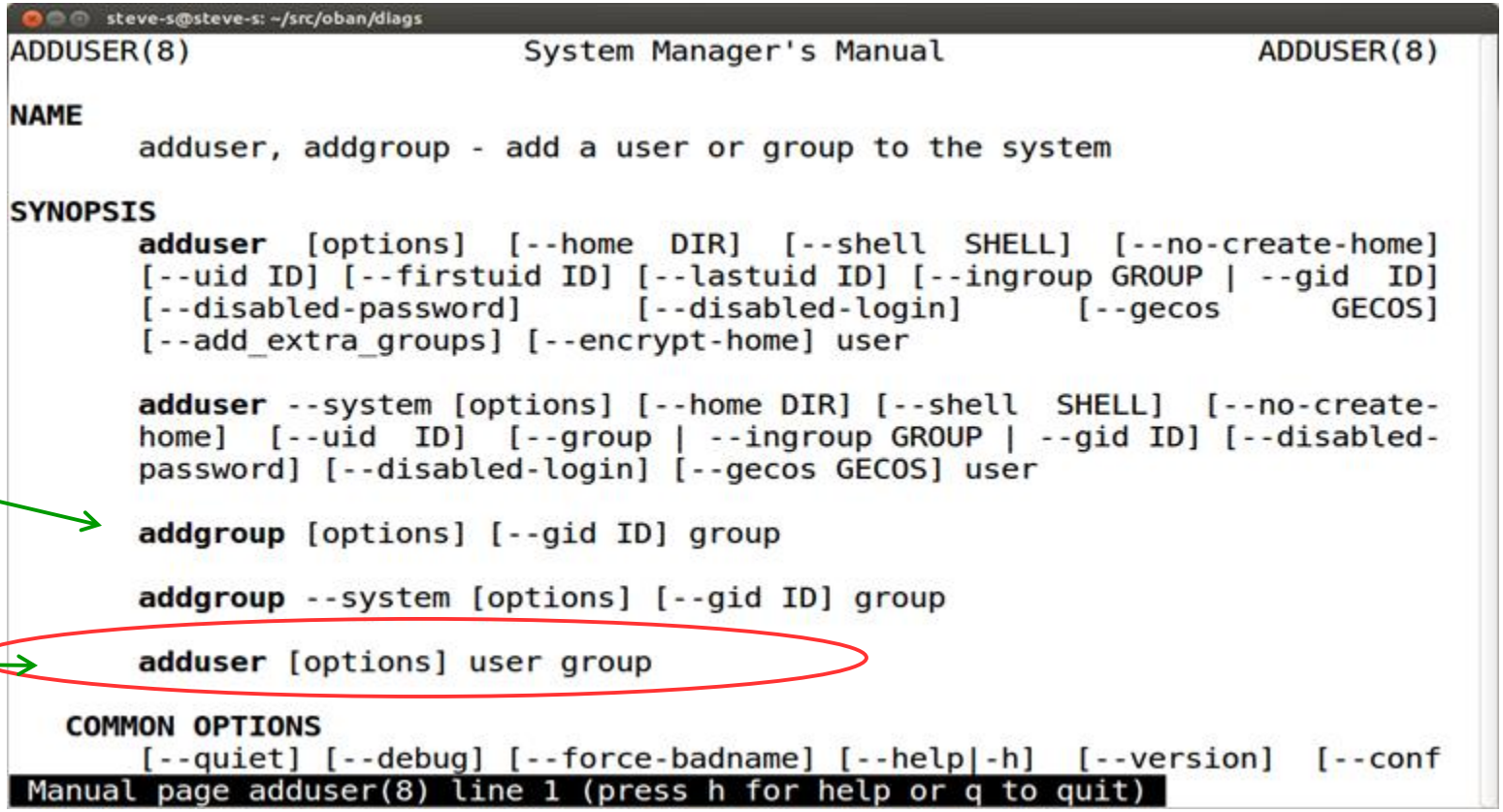
- Whoa! This is confusing.
- Make a user with $ useradd command and compare the difference.

Add a user to an existing group

- Edit the group control file? (What is it?)
- Use the `$ adduser` command with the two arguments: the user, followed by the group.
- Need to use `sudo`, of course.

Create *new* group

Add user to *existing* group



adding multiple users

- Use `newusers` script to add *a lot* of users at once.
- A Sys Admin might use this script in *another* script which also emailed a 'welcome' script to an email address.

```
File  Edit  View  Terminal  Help
NEWUSERS(8)                    System Management Commands                    NEWUSERS(8)

NAME
       newusers - update and create new users in batch

SYNOPSIS
       newusers [options new_users]

DESCRIPTION
       The newusers command reads a file of user name and clear-text password pairs and uses this
       information to update a group of existing users or to create new users. Each line is in the
       same format as the standard password file (see passwd(5)) with the exceptions explained
       below:

       pw_name:pw_passwd:pw_uid:pw_gid:pw_gecos:pw_dir:pw_shell

       pw_name
           This is the name of the user.

           It can be the name of a new user or the name of an existing user (or an user created
 Manual page newusers(8) line 1
```

The Superuser: Root

- Every Linux system has a user called '`root`'
- The root user is *all-powerful*
  - Can access any files - system and other users.
- The root user account should only be used for system administration, such as installing software
- When logged in as root, the shell prompt usually ends in #

```
$ whoami
fred
$ su -
Password:    (root password)
# whoami
root
```

sudo

- https://help.ubuntu.com/community/RootSudo
- By default, the root account password is *locked* in Ubuntu.
- This means that you cannot login as root directly or use the su command to become the root user.
- However, the root account still physically exists
- And yet we still need to run as root to administer the system
- This is where `sudo` comes in. It allows authorized users to run certain programs as root without having to know the root password.
- That is, sudo checks the credentials of the *actual* user, and if allowed, will then allow that user to run a command as root.

```
$ sudo chown steve lab7.txt
[sudo] password for steve:
$
```

sudo advantages

- Root password is not *vulnerable*
  - Never typed in over a network connection
  - Never written down
  - Never put in a file on a disk
  - Never on someone's smart phone
- Only run 1 command.
  - It's very easy to log into a root shell, forget you are there, and cause damage. This wouldn't have happened if not root.

Unix

```
$ su -
Password:
# chmod 644 <file>
…
# rm -rf
```

Ubuntu

```
$ sudo chmod 644 <file>
[sudo] password for steve:

$ rm -rf
Don't have permissions.
```

Getting a root shell

- There are still times when you need to get a root shell.
- You may have a piped command, and it needs root access – it's easiest to carefully use a root shell
- In Ubuntu's methodology, you can't "su -" because you don't know the root passwd. Instead, use "`$ sudo -s`"
  - Think of -s option as "shell"

Unix

```
$ whoami
fred
$ su -
Password:        (root pw)
# whoami
root
```

Ubuntu

```
$ su -
Password:   (root pw)
su: Authentication failure
$ sudo -s
[sudo] password for steve:    (my pw)
#
```

How to setup sudo

- IDs which *can* sudo are called “sudoers”

```
$ man sudoers
```

- There is a sudo group
  - It's set up during Ubuntu install process
  - gid 27 on my system
- You can simply add an existing user to the sudo group

```
$ sudo adduser <username> sudo
```

sudo control file

- sudo policy is controlled in file `/etc/sudoers`
- Because it's so important and to make sure you edit `/etc/sudoers` correctly, there is a *preferred method* to change sudoers.

```
$ man visudo
```

```
$ cat /etc/sudoers
/etc/sudoers: Permission denied

$ sudo cat /etc/sudoers
[sudo] password for steve:
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
```

A complication of using sudo → redirect

- Since `sudo` is the first command, *redirecting* the output of the `sudo` command requires some thought.
- For instance, consider

```
$ sudo ls > /root/somefile
```

will not work since it is the non-sudo shell that tries the re-direct, *i.e.* to write to that protected file in /root

- Solutions:
  1. You can use `$ ls | sudo tee /root/somefile` to remove the > (redirect)
  2. Run a shell process run as root:

```
$ sudo sh -c "ls > /root/somefile"
$ sudo sh -c "cd /home ; du -s * | sort -rn"
```

Graphical sudo

- Use `gksu` or `gksudo` to run a graphical program with a GUI.

```
$ gksu gedit /etc/fstab
```

Switching users

- Use `su` to switch to a different user
  - Quicker than logging off and back on again
- `su` prompts you for *that user's* password:

```
$ su - bob
Password:
```

  - The - option makes su behave as if you've logged in as that user
  - *i.e.,* cd's to that dir, runs the `.profile`, etc.
  - When done as that user, ^D to exit that shell
- Infrequently, the root may want to "become" that user:

```
$ sudo su - bob
```

change a passwd

- To change your password, use `$ passwd` command
  - You have to know the old one!
- To change *another* user's password, you use

  `$ sudo passwd bob`
  - It does not ask for the old password! It just changes it!!!

For user fred:

```
$ passwd
Changing password for fred.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
```

For another user with `sudo` access

```
$ sudo passwd fred
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```